SENTENCE-HYPOTHESES GENERATION IN A CONTINUOUS-SPEECH RECOGNITION SYSTEM

Volker Steinbiss

Philips GmbH Forschungslaboratorium Hamburg,
D-2000 Hamburg 54, P.O. Box 540840, FRG

ABSTRACT- In this paper, the dynamic-programming algorithm for continuous-speech recognition is modified in order to obtain a top-N sentence-hypotheses list instead of the usual one sentence only. The theoretical basis of this extension is a generalization of Bellman's principle of optimality. Due to the computational complexity of the new algorithm, a sub-optimal variant is proposed, and experimental results within the SPICOS system are presented.

0 INTRODUCTION

The output of a continuous-speech recognition system using Viterbi decoding is the word sequence associated with the most likely hidden Markov model state sequence. In the continuous-speech database query and answering system SPICOS, this word sequence is passed to a language-interpretation module. Subsequently, data-base query evaluation, answer generation and speech synthesis have to be carried out in order to produce a spoken answer.

If, in the case of a speech recognition error, the recognized sentence cannot be parsed or interpreted, the system may have to stop at this point as no further information is available from the recognition module. So, it is desirable that the speech-recognition module does not pass one sentence only but a list of sentences instead which are most likely to have been uttered. For the level-building algorithm, an approach to the top-N decoding problem has been proposed in [5].

The subject of this paper is to modify the one-stage dynamic-programming search algorithm in order to produce a top-N list of sentence hypotheses. The memory and computation-time requirements of the exact approach seem to make its use in a system rather difficult. Therefore, in chapter 2, a sub-optimal approach with only moderate consumption of storage and processing time is described. The performance of this approach is subject to the experiments discussed in section 3.

1 A GENERALIZATION OF BELLMAN'S PRINCIPLE OF OPTIMALITY TO TOP-N WORD SEQUENCES

1.1 Bellman's Principle of Optimality: The Optimal State Sequence

Viterbi decoding in connected-speech recognition solves the following problem: Given a hidden Markov model (HMM) ([1], [6]), find the state sequence which most probably has produced the observed input $Y = y(1), ..., y(T)$. The word sequence W associated with this state sequence might be sub-optimal in the sense that it does not maximize the likelihood $p(W,Y)$ ([1]); nevertheless good results have been obtained with Viterbi decoding, and it is also used within the SPICOS system (cf. section 3).

The problem can be reformulated as to find a state sequence (also called 'path') which minimizes an additive cost function:

Over all paths $s(0), ..., s(T)$ from a beginning state $s_B = s(0)$ to a final state $s_F = s(T)$, minimize

$$\sum_{t=1}^{T} c(t, s(t-1), s(t))$$

where $c(t, i, j)$ are the costs of moving from state i at time t-1 to state j at time t. This cost function depends on the underlying model: In the HMM approach, it is the negative logarithm of the probability of going from state i at time t-1 to state j at time t and of producing the output y(t). As in the dynamic time warping (DTW) approach, it might just as well be some distance between a reference vector and the observed vector y(t), plus a time distortion penalty. We are not limited to either of these interpretations but would like to describe, as generally as possible, the dynamic-programming (DP) recursions (this section) as well as their generalization to generating top-N hypotheses lists (sections 1.2. and 1.3.).

In order to find the optimal state sequence through S states there is no need to evaluate all of the S^{T-1} possible paths. Indeed, Bellman's simple but powerful *principle of optimality* ([2]) states that if the optimal state sequence from s(0) to s(T) passes through state s(t) at time t, then it includes, as a portion of it, the optimal state sequence from s(0) to s(t). In other words, a non-optimal partial path can never be part of the global optimal path.

Denote by $P(t,s) = s(0), ..., s(t)$ the optimal path from $s_B = s(0)$ to state $s = s(t)$ (or *an* optimal path; for conciseness, we will not consider non-uniqueness here nor do we define the case when there is no path to $s = s(t)$ ('infinite costs')), and denote its accumulated costs by $C(t,s)$. The global optimal path $P(T,s_F)$ from beginning state s_B to final state s_F and its accumulated costs $C(T,s_F)$ can be calculated recursively by

Initialization:

$$C(0,s) = \begin{cases} 0 & : s = s_B \\ \text{infinity} & : \text{else} \end{cases} \qquad (1)$$

$$P(0,s) = \begin{cases} s_B & : s = s_B \\ \text{empty} & : \text{else} \end{cases} \qquad (2)$$

Recursion over $t = 1, ..., T$:

$$C(t,s) = \min_{s'} (C(t-1,s') + c(t,s',s)) \qquad (3)$$

$$P(t,s) = P(t-1,s')\circ s \quad \text{is the state sequence obtained by concatenating } P(t-1,s') \text{ with } s \text{, where } s' \text{ is the predecessor state that minimizes (3)} \qquad (4)$$

Fig. 1 illustrates how the optimal path $P(T,s_F)$ is obtained by successively following the minimizing arguments s'.

Note that, in this general case, the complexity of calculating $C(T,s_F)$ drops from S^{T-1} to $S\cdot(T-1)$.

1.2 Top-N State Sequences

The optimality principle is generalized from seeking the optimal state sequence to seeking the top-N state sequences simply as follows: If, for $n = 1, ..., N$, the n-th best state sequence from s(0) to s(T) passes through state s(t) at time t, then it includes, as a portion of it, the m-th best (with some $m \in \{1, ..., n\}$) state sequence from s(0) to s(t). In other words, a partial path not among local top-n can never be part of the global n-th best path.

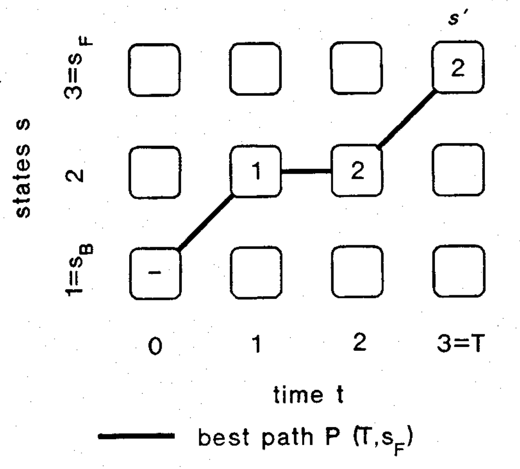

Fig. 1: In standard dynamic-programming search, the optimal path $P(T,s_F)$ is obtained by successively following the values for s' that minimize formula (3).

Let $P_n(t,s)$ denote the n-th best path from $s_B = s(0)$ to $s = s(t)$ and let $C_n(t,s)$ denote its accumulated costs. Then the recursion is:

Initialization:

$$C_n(0,s) = \begin{cases} 0 & : n = 1 \text{ and } s = s_B \\ \text{infinity} & : \text{else} \end{cases} \qquad (5)$$

$$P_n(0,s) = \begin{cases} s_B & : n = 1 \text{ and } s = s_B \\ \text{empty} & : \text{else} \end{cases} \qquad (6)$$

Recursion over t = 1, ..., T:
Let, for n = 1, ..., N,

$$C_n(t,s) = \min_{n',s'} (C_{n'}(t-1,s') + c(t,s',s) \mid P_{n'}(t-1,s')\circ s \neq P_m(t,s) \text{ for all } m<n). \qquad (7)$$

(I. e., the minimum is taken over those state sequences, consisting of a specified partial path and state s = s(t), which are not yet in the local top-(n-1) list. Note that there is no restriction for n = 1.)

$$P_n(t,s) = P_{n'}(t-1,s')\circ s \text{ , where } n',s' \text{ minimize (7).} \qquad (8)$$

Fig. 2 illustrates how to obtain the n-th best path.

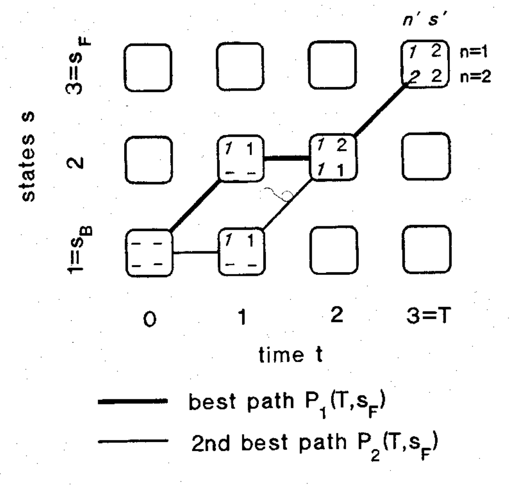

Fig. 2: This figure illustrates how, for N=2, the top-N paths are obtained by successively following the values for n' (position in local top-N list) and s' (preceeding state) that minimize formula (7).

1.3 Top-N Best Word Sequences

A top-N list of state sequences is not yet what we want - it might result from different time alignments of only one word sequence. In order to get a list of top-N word sequences one must, during recombination, take into account for each state sequence P the word sequence W(P) associated with P. Accordingly, while the initialization remains as (5) and (6), the recursion formulae have to be changed as follows:

Recursion over t = 1, ..., T:
Let, for n = 1, ..., N,

$$C_n(t,s) = \min_{n',s'} (C_{n'}(t-1,s') + c(t,s',s) \mid W(P_{n'}(t-1,s')\circ s) \neq W(P_m(t,s)) \text{ for all } m<n) \qquad (9)$$

(The minimum is taken over those state sequences whose associated word sequence is not yet in the local top-(n-1) list.)

$$P_n(t,s) = P_{n'}(t-1,s')\circ s \text{ , where } n',s' \text{ minimizes (9).} \qquad (10)$$

Eventually, the same optimal state sequence $P_1(T,s_F)$ and word sequence $W_1 := W(P_1(T,s_F))$ are obtained as in the preceeding sections. In contrast to the preceeding section, $P_2(T,s_F)$ here is the optimal path among all paths with an associated *word sequence* different from W_1, etc.

2 A SUB-OPTIMAL VARIANT OF TOP-N DYNAMIC PROGRAMMING

Using the recursion formulae (5), (6), (9) and (10), the usual one-stage dynamic-programming algorithm (cf. [12], [3], [7]) can be modified in order to obtain a top-N sentences hypotheses list instead of one sentence only. For an efficient implementation, the following seems to be important:

- A fast access to the word-sequence list associated with a point in the search space. This can be realized by constructing a tree containing all possible partial word sequences so far seen and handling only its node labels within the recombination.
- A fast sorting or merging procedure for the recombination of top-N lists.

As the search space is rather big - though using a pruning strategy, in SPICOS several thousand points have to be processed for every centisecond of speech - the extra effort for getting top-N sentence hypotheses instead of one sentence seems to be rather high. This relates to the computational costs for the recombination as well as to the storage of top-N lists. Therefore, we propose a sub-optimal variant of top-N DP by restricting the extended recombination (formulae (9) and (10)) to the syntax level and letting the in-word recombinations be the standard ones (formulae (3) and (4)). The effects of this clearly sub-optimal strategy are discussed below. Its advantage with respect to the exact top-N DP search is the reduced computational and storage complexity, because most interactions are in the word interiors. Experimental results are presented in section 3.

The proposed approach ignores the fact of in-word recombinations: It does not obtain the exact solution, i. e. the best path giving the n-th best word sequence, if and only if this path recombines somewhere inside a word with a better scored path. Whereas on the syntax level all necessary information about the (locally) top-n word sequences is stored, only best decisions are regarded inside words (on the state level). That means the algorithm finds a sub-optimal solution only.

To see in more detail what happens in such a case, regard fig. 3 as an example which shows a part of the search space. Inside any of the words u, v, w, the HMM is left-to-right; from the word ending states, every word beginning state can be reached. Imagine that uw and vw are the best and second best word sequences, resp. Three paths, each of which optimal in a certain sense, are subjects of our interest:

- The global optimal path P_1 (through word sequence uw)
- The best path P_2 through the second best word sequence vw. It is lost (i. e. unavailable for trace-back) because it recombines inside word w with the better scored path P_1.

The best path through vw which is available for trace-back is

- path P_3, which is the best path through vw satisfying the additional requirement that the word boundary between v and w coincides with the word boundary between u and w.

Generally, P_2 will have a better score than P_3, that is, the path P_3 found is only sub-optimal. On the other hand, if P_2 recombines with P_1 inside w, there is some indication that P_2 and P_3 are relatively close to each other anyway and that so are their costs. This can be taken as a justification for using this sub-optimal decoding strategy in a system. The experimental results of the following chapter show indeed that although the sub-optimality is visible, the proposed approach solves the top-N decoding problem with sufficient accuracy.

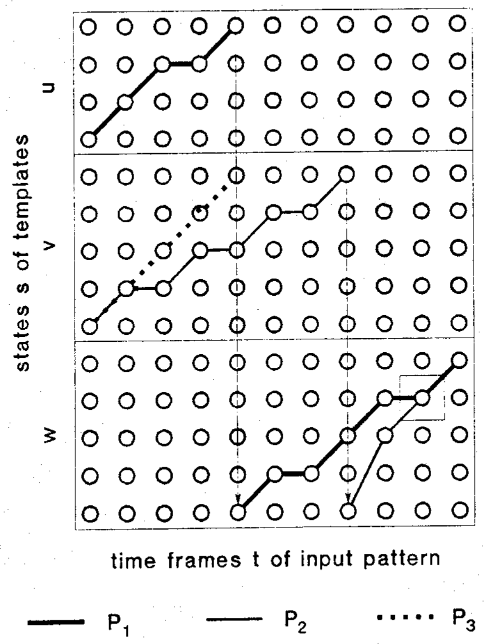

Fig. 3: Illustration of the approximation error. The best path P_2 for word string vw recombines with the globally best path P_1 (word string uw) inside word w and is thus unavailable for trace back. Only the path P_3 through uw which passes the syntax node at the same time as the path through vw (i. e., the word boundaries are reached at the same time) can be recovered by trace-back.

3 EXPERIMENTAL RESULTS

3.1 The Recognition System Environment

Experimental tests were performed within the SPICOS system, with 'SPICOS' standing for 'Siemens-Philips-IPO continuous-speech recognition and understanding system'. Within this joint project a man-machine dialogue system has been developed which provides voice access in German to a data base containing information about the project itself. The SPICOS system consists of modules for speaker-dependent continuous-speech recognition, language interpretation, database-query evaluation, answer generation, and speech synthesis. For details, cf. [8], [9], [10], [11].

The phoneme-based large-vocabulary speech-recognition module of SPICOS as used for the tests described below can be characterized as follows:

- Office environment, close-talking microphone.

Preprocessing:

- Sampling rate 16 kHz
- 30 cepstrally smoothed spectral intensities in logarithmic units, normalized with respect to average intensity, plus intensity
- Additionally, first and second differences of these.

Acoustic-phonetics:

- Standard pronunciation dictionary (one pronunciation per word)
- 44 context-independent phonemes
- Continuous mixture-density distributions.

Search:

- Data-driven one-stage DP search, modified according to chapter 2 in order to obtain (sub-optimal) top-N sentence hypotheses lists.

Language model:

- Recognition vocabulary comprising 917 words
- Either finite state grammar (test set perplexity ([4]) 85) or stochastic bigram language model (test set perplexity 124).

Training and test data:

- 2 times 100 phonetically balanced training sentences (7 minutes of speech)
- 200 test sentences (data-base queries)
- Small overlap only of training and recognition vocabularies.

3.2 Recognition Results

For each of four speakers (two male, two female), recognition tests have been performed on 200 test sentences. Table 1 shows, for N from 1 to 10 and the two language models, how often the spoken sentence is included in the top-N sentence hypotheses list. For the finite-state language model, more than one half of the spoken sentences which were not correctly recognized (i.e., not in position 1) was found among top 4. On the other hand, one third was not even among top 12. For the bigram language model, one third of the incorrectly recognized sentences can be found among top 5. For completeness, the top-N error rates are given in table 2, defined as deletions + insertions + substitutions of (for each spoken sentence) the best recognition in the top-N list, divided by the number of spoken words. Both top-N sentence recognition rate and top-N error rate show what at best could be achieved by post-processing the sentence-hypotheses lists with a powerful parser; in some cases, however, not even human knowledge would be sufficient to recover the spoken sentence from a list.

A first experiment with the top-12 lists of one speaker (F-10; finite-state language model) showed that among those 158 of the 200 spoken test sentences which can be correctly interpreted by the SPICOS parser (cf. [11]), the best sentence which is accepted by the parser is in 137 cases the spoken one, while only 123 of these sentences are in top-1 position: So some extra effort in the search procedure reduced the number of not recognized sentences by a third.

For a few spoken sentences, additional time alignments have been carried out for each sentence on the top-10 list. The results were as could be expected from theory: In some of the cases, the best sentence (position 1) forced its word boundaries to the poorer scored candidates, thus often increasing their scores with respect to the exact ones: The effect of recombining the best path of a word sequence with a better path of another word sequence described in section 2. The exact scores generally forced a reordering of the sub-optimal top-10 list. Nevertheless, these effects are small enough to be tolerated in a real system.

3.3 Computational and Storage Costs

The storage costs of top-N DP are proportional to N, the length of a list. In our system, the storage for trace-back pointers etc. increased by a factor of 2N; the additional factor of 2 is due to the fact that besides the usual one transition number and one back pointer, the cost difference to the top hypothesis and a pointer to the in-list position have to be stored.

In order to get experimental results about the computational costs of the algorithm, five off-line recognition tests (with finite-state grammar) were performed on the 200 SPICOS sentences (754 seconds of speech) for the same (female) speaker. The word error rate is 4.7 % ((word substitutions + deletions + insertions) / number of words), the average number of active grid points per test frame is 2344, and (at least) 4 optimal paths are lost by the search procedure due to pruning.

The result is illustrated in table 3 which shows that the extra computational cost for recovering top-N sentences, compared to the usual search, is significant for high N (here: N = 8) only but negligible for small ones. Indeed, the figures show that the measured CPU times, which depend on the load of the machine, are only a rough clue for the computational costs, which increase strictly from 'STANDARD' to 'TOP 8' because the computations of each test job are a subset of the following one.

Nevertheless, we can conclude that extended trace back for top N can be used for reasonable values of N (e.g., N = 5) without significant effects on recognition time. For high values of N, the list sorting becomes more and more important, such that a more sophisticated implementation is needed.

SUMMARY

Based on a generalization of Bellman's optimality principle, the one-stage dynamic-programming search algorithm can be modified in order to find the top-N sentences with respect to the Viterbi criterion. A sub-optimal variant of the top-N dynamic-programming search has been presented and tested within the SPICOS system.

Experimental tests yielded quite satisfying results in two aspects: For the finite-state grammar, the majority of the sentences not correctly recognized are at least among top 4, and the deviation from the exact solution is tolerable. For reasonable values of N (e.g., N = 5), the additional storage and computational amount is small with respect to the other SPICOS requirements. Thus, it has been demonstrated that a continuous-speech recognition system based on the one-stage dynamic-programming algorithm can be modified in order to produce a top-N sentence-hypotheses list with little effort only on the level of the book-keeping for trace-back.

ACKNOWLEDGMENT

The work described was carried out in a joint Siemens - Philips - IPO (Eindhoven) project ('SPICOS') and partially funded by the German Federal Ministry for Research and Technology (BMFT) under grant No. 413-5839-ITM 8801A. Only the author is responsible for the content of this publication.

REFERENCES

[1] L. R. Bahl, F. Jelinek, R. L. Mercer, "A Maximum Likelihood Approach to Continuous Speech Recognition", IEEE Trans. on Pattern Analysis and Machine Intelligence, Vol. PAMI-5, No. 2, pp. 179-190, March 1983.
[2] R. Bellman, "Dynamic Programming", Princeton, NJ: Princeton Univ. Press, 1957.
[3] J. S. Bridle, M. D. Brown, R. M. Chamberlain, " An Algorithm for Connected Word Recognition", Proc. 1982 Conf. on Acoustics, Speech and Signal Processing, Paris, France, pp. 899-902, May 1982.
[4] F. Jelinek, "The Development of an Experimental Discrete Dictation Recognizer", Proc. IEEE, vol. 73, pp. 1616-1624, 1985.
[5] C.-H. Lee, L. R. Rabiner, "A Network-Based Frame-Synchronous Level Building Algorithm For Connected Word Recognition", Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing, New York, pp. 410-413, 1988.
[6] S. E. Levinson, "Structural Methods in Automatic Speech Recognition", Proc. IEEE, vol. 73, No. 11, pp. 1625-1650, 1985.
[7] H. Ney, "The Use of a One-Stage Dynamic Programming Algorithm for Connected Word Recognition", IEEE Trans. Acoust., Speech, Signal Processing, vol. ASSP-32, pp. 263-271, Apr. 1984.
[8] H. Ney, D. Mergel, A. Noll, A. Paeseler, "A Data-Driven Organization of the Dynamic Programming Beam Search for Continuous Speech Recognition", Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing, Dallas, TX, pp. 20.10.1-4, April 1987.
[9] H. Ney, A. Paeseler, "Phoneme-Based Continuous Speech Recognition Results For Different Language Models in the 1000-Word SPICOS System", Speech Communication 7, pp. 367-373, 1988.
[10] A. Paeseler, "Continuous-Speech Recognition Using a Stochastic Language Model", Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing, Glasgow, pp. 719-722, May 1989. .
[11] G. Thurmair, "Semantic Processing in Speech Understanding",in: Recent Advances in Speech Understanding and Dialog Systems, H. Niemann et al. (eds.), NATO-ASI Conference, July 1987: Springer-Verlag Berlin Heidelberg, pp. 397-419, 1988.
[12] T. K. Vintsyuk, "Element-wise recognition of continuous speech composed of words from a specified dictionary", Kibernetika, vol. 7, pp. 133-143, Mar.-Apr. 1971.

	speaker					
top N	F-01	F-10	M-03	M-10	total	/ in %
1	133	157	164	137	591	73.9
2	161	172	183	148	664	83.0
3	171	176	187	158	692	86.5
4	175	182	191	159	707	88.4
5	178	184	191	160	713	89.1
6	178	186	192	161	717	89.6
7	181	187	192	162	722	90.3
8	184	187	192	165	728	91.0
9	184	188	192	165	729	91.1
10	185	188	192	165	730	91.3
total	200	200	200	200	800	100.0

Table 1a: Correct sentences among top-N sentence lists. N = 1 is "sentence correctly recognized". Language model: finite-state syntax (perplexity 85).

	speaker					
top N	F-01	F-10	M-03	M-10	total	/ in %
1	106	139	143	101	489	61.1
2	132	152	162	113	559	69.9
3	136	156	166	118	576	72.0
4	140	159	169	119	587	73.4
5	140	162	171	120	593	74.1
6	143	165	171	121	600	75.0
7	145	167	172	125	609	76.1
8	146	170	173	127	616	77.0
9	148	171	175	127	621	77.6
10	149	172	176	128	625	78.1
total	200	200	200	200	800	100.0

Table 1b: Correct sentences among top-N sentence lists. N = 1 is "sentence correctly recognized". Statistical language model (bigram, test set perplexity 124).

	speaker				
top N	F-01	F-10	M-03	M-10	average
1	11.5	5.7	5.5	13.7	9.1
2	8.7	4.3	3.5	11.8	7.1
3	7.9	4.0	3.1	11.0	6.5
4	7.5	3.7	2.7	10.6	6.1
5	7.3	3.5	2.6	10.2	5.9
6	7.0	3.2	2.6	10.0	5.7
7	6.8	3.0	2.5	9.4	5.4
8	6.5	2.7	2.4	9.1	5.2
9	6.3	2.7	2.2	9.0	5.1
10	6.1	2.6	2.1	8.8	4.9

Table 2: Top-N error rates (defined in section 3.2) in % (bigram language model used).

METHOD	CPU TIME	IN SECONDS	× REAL TIME	RELATIVE
STANDARD	03:55:03.98	14 104	18.7	100.0 %
TOP 1	03:44:55.55	13 496	17.9	95.7 %
TOP 2	03:44:02.15	13 442	17.8	95.3 %
TOP 4	04:00:54.45	14 454	19.2	102.5 %
TOP 8	04:57:08.66	17 829	23.6	126.4 %

(TESTS PERFORMED ON A VAX 8700 - 6 MIPS)

Table 3: Effect of the number of top sentence hypotheses on the computation time of SPICOS (with finite state syntax).